## REMARKS

Reconsideration and allowance are respectfully requested.

Applicants acknowledge with appreciation the indication that claims 5 and 12 are allowable.

Claims 1-4, 6-11, and 13-15 stand rejected under 35 U.S.C. 103 as being obvious over USP 4,747,040 to Blanset in view of USP 5,515,538 to Kleiman. This rejection is respectfully traversed.

The inventors in this case recognized a problem that can arise when a request for further processing is received by an operating system which has been suspended as a consequence of interrupt. The operating system which has been controlling processing in place of the suspended operating system may have performed a "task switch", or another event may have occurred, which results in the processing request sent to the suspended operating system being a request to resume or start a thread or task *different* from that which was active in the suspended operating system when it was suspended. This problem tends to arise in the context of secure and non-secure operating systems where there is a limited ability for the operating systems to exchange information without compromising security. The problem may arise in other contexts as well such as in the support of legacy operating systems running with other operating systems where the legacy operating system does not have any provision for the exchange of information with another operating system to enable co-ordination as to thread switching.

The inventors solved this problem by restarting the suspended operating system with a return interrupt which triggers execution of return interrupt handling code by the now-reactivated operating system. The return interrupt handling code can identify from the return interrupt provided by the currently active operating system the particular data processing thread being

requested. If this requested (return) processing thread is the same as the active thread when the now-resumed operating system was suspended, then this tread may be resumed. But if the return data processing thread is not the same as the suspended data processing thread, then the return interrupt handling code can trigger or perform a thread switching operation to save context data associated with the original suspended thread so it can later be restarted and either start the newly requested thread or resume such a requested thread if it had already been started earlier in the processing operations. The solution can be thought of as delivering the interrupt to both operating systems. The original operating system is suspended, and the interrupt is delivered to the other operating system to be handled. Then, an interrupt is delivered by the other operating system to the suspended operating system to warn the suspended operating system of any task switch that might have occurred while it was suspended.

Claims 1 and 8 are amended to more clearly recite "determining under control of said second operating system a return data processing thread to be executed under control of said first operating system." Example support for this amendment can be found in Figure 26 and in the description of Figure 26 on pages 52-53. This text specifies how, starting from a non-secure mode of operation (an example of a second operating system) at step 13, the secure mode of operation (an example of a first operating system) is re-entered at step 15 using a software-faked return interrupt from the non-secure operating system which specifies a return thread ID determined by the non-secure operating system. The return thread ID identifies the thread "requested by the non-secure thread NSB" to be started by the secure operating system (first operating system) at step 15. The suspended secure operating system uses that return interrupt to trigger execution of return interrupt handling code by the reactivated sure operating system. The return interrupt generated by the non-suspended operating system identifies the appropriate data

processing thread to be executed under control of the suspended secure operating system upon reactivation.

Neither Blanset nor Kleiman discloses or suggests "determining under control of said second operating system a return data processing thread to be executed under control of said first operating system" (quoted from claim 1). The Examiner refers to the description of Figure 14 of Blanset, but that description actually *teaches away* from the quoted feature from amended claim 1. Column 16, lines 15-28 describes how the Blanset system deals with an interrupt. In Figure 14, the system operate under control of a UNIX operating system. A UNIX interrupt is received at stage 1402, which causes control to pass to an MS-DOS operating system at stage 1407, whereupon "a return routine is then invoked, as indicated at block 1406, and execution resumes at the point at which the computer was interrupted."

In this example, the UNIX operating system is considered to be the first operating system an the MS-DOS operating system the second operating system. Blanset teaches that the UNIX operating system simply resumes processing of the interrupted thread. Figure 14 of Blanset does not even consider the possibility that a "thread switch" may have occurred prior to returning from the MS-DOS operating system to the UNIX operating system. Blanset neither discloses nor suggests that the MS-DOS operating system determines the specific return data processing thread to be executed under control of the UNIX operating system on return from the interrupt.

Returning to the non-limiting example from Figure 26 of the instant application, the second operating system (e.g., the non-secure operating system in Figure 26) determines the return data processing thread to be executed under control of the first operating system (e.g., the secure operating system of Figure 26) such that in the case where a thread switch has in fact occurred (see stage 11 of Figure 26), then execution of the second data processing thread (i.e.

thread B in the Figure 26 example) will be resumed by the return interrupt handler at stages 15, 16 and 17, rather than resuming execution of thread A where the process was interrupted (see stages 5 and 6). The non-limiting, example embodiment of Figure 27 shows that if there is no thread switch such that the return data processing thread is still thread A, then the second operating system (non-secure) determines that the return data processing thread on re-activation of the secure mode should be the thread A.

Even if Klieman were combined with Blanset, that combination still fails to teach all the features recited in the independent claims 1 and 8 because neither discloses determining "under control of said second operating system a return data processing thread to be executed under control of said first operating system." Accordingly, the application is in condition for allowance. An early notice to that effect is earnestly solicited.

Respectfully submitted,

**NIXON & VANDERHYE P.C.**

By: _____
John R. Lastova
Reg. No. 33,149

JRL:maa
901 North Glebe Road, 11th Floor
Arlington, VA 22203-1808
Telephone: (703) 816-4000
Facsimile: (703) 816-4100